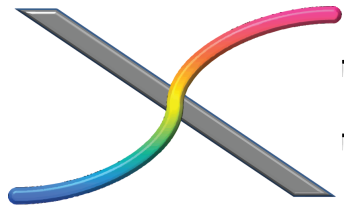


EXCESS

Pruning Strategies in Adaptive Off-line Auto-tuning

Lu Li, Usman Dastgeer, Christoph Kessler
Linköping University

This project is part of the portfolio of the
A.3 – Advanced Computing and Complex System Unit
Communications Networks, Content and Technology DG
European Commission

www.excess.eu

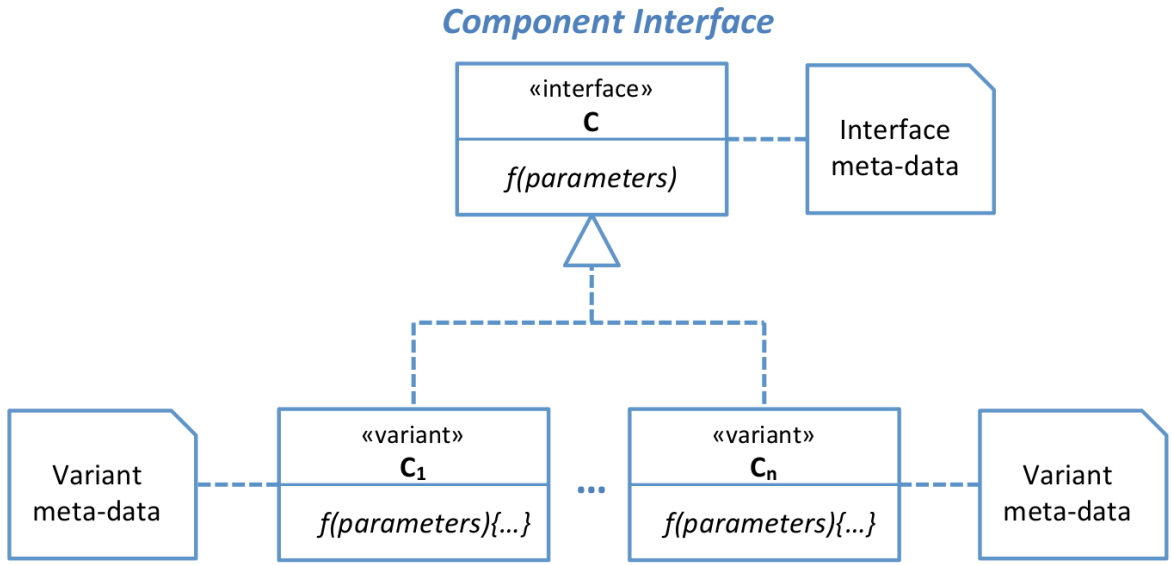
Contract Number: 611183
Total Cost [€]: 3.31 million
Starting Date: 2013-09-01
Duration: 36 months

Agenda

Problem Background

- Application needs tuned for optimal performance
- Performance tuning is challenging
  - Heterogeneous processors
  - Configuration diversity
- Auto-tuning needed
  - Performance Portability
- Approach: Implementation selection for a multi-variant component

Component Interface and Implementations



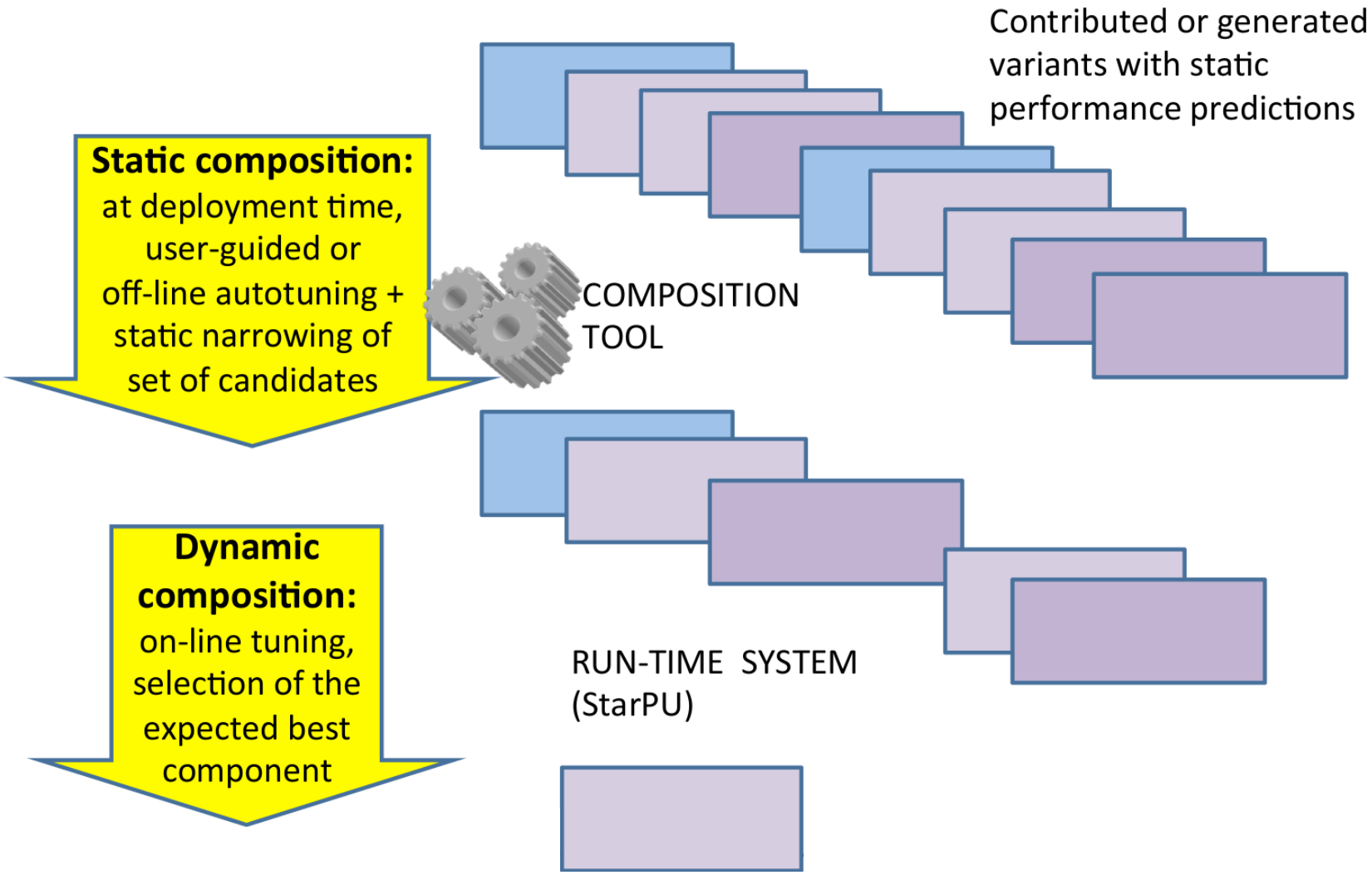
Implementation variant

- Platforms (CPU, accelerator cores)
- Algorithms
- Tunable parameter settings
- Compiler transformations
- ...

Meta-data

- Dependencies
- Resource requirements
- Deployment descriptors
- Performance prediction models
- ...

Staged Composition



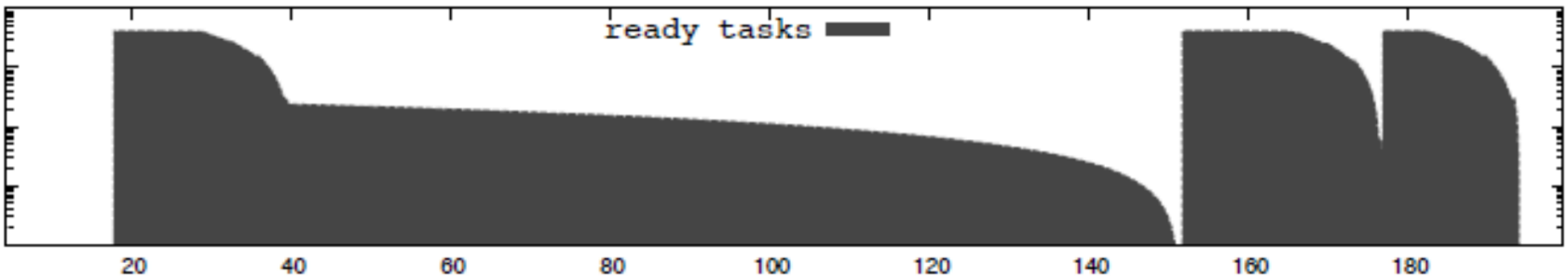
Guiding Composition: Empirical Models

- Analytical model
- Empirical model
  - No or little understanding of the target architecture
  - Off-line Empirical Models: Feed sampling data (training examples) into a prediction model, e.g. SVM, C4.5

Off-line Empirical Models: Pros and Cons

- + Avoid "cold start"
- + Controllable tuning process
- - Tuning overhead

"Cold start" Effect



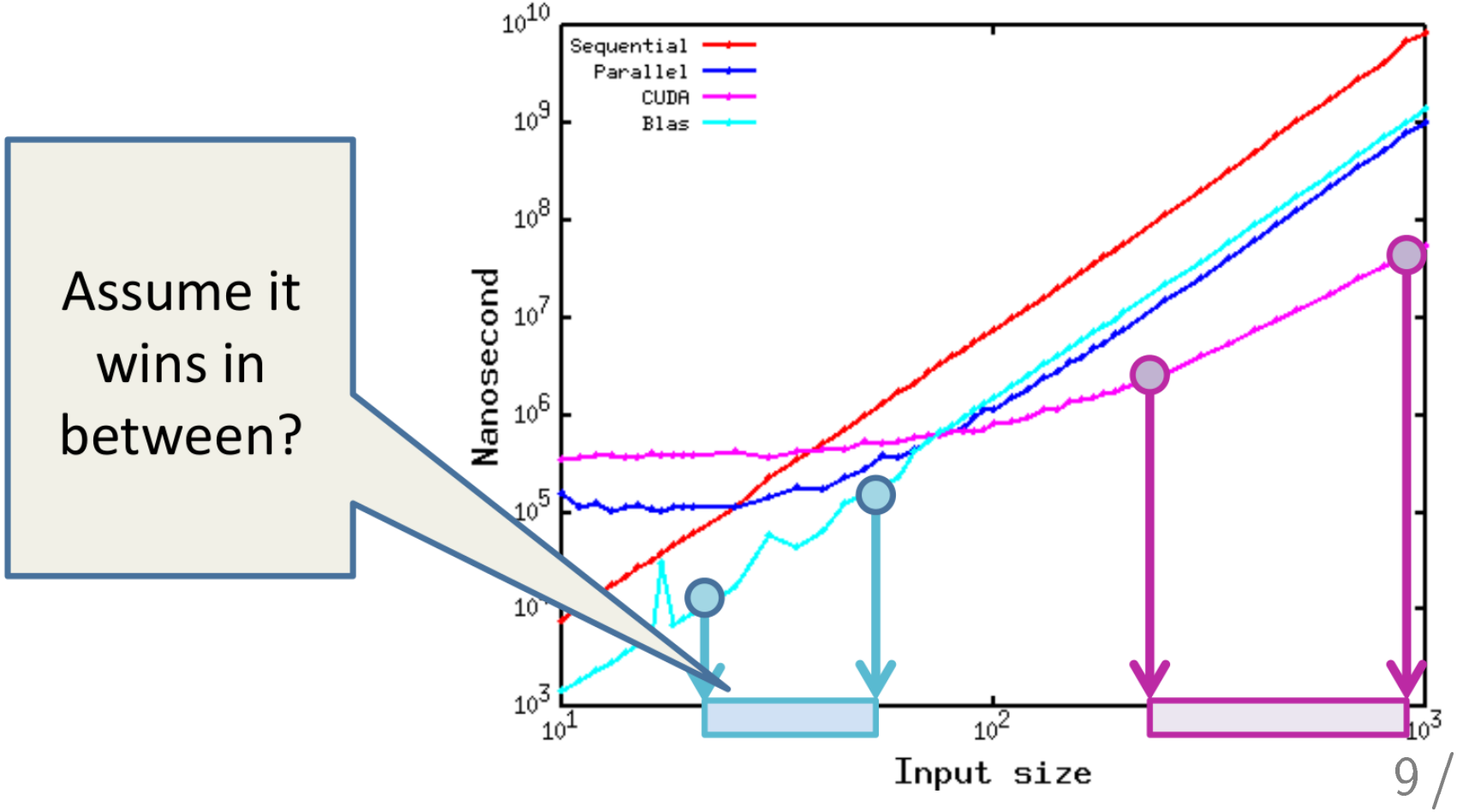
Example from Dastgeer et al. (ParCo'2011) on SkePU/StarPU integration, Coulombic potential grid execution, with 3 successive executions

Zoom into the Cons

- A closer look at off-line overhead
  - Exhaustive execution is not feasible
  - Previous work: Stargazer (random sampling)
- Training examples (Sampling data) is vital for performance prediction models

Attack the Cons: Observations

- In the context of performance tuning: a concrete example (Matrix-matrix multiplication)
- Smart sampling by heuristics



Adaptive Sampling

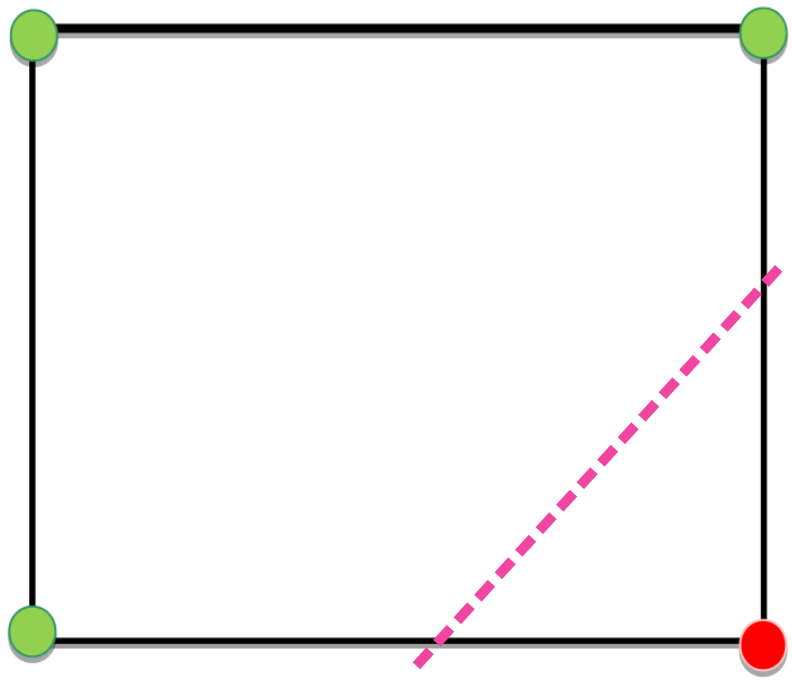
- Sample only vertices of a space
- Recursive decomposition (cutting) for open spaces, controlled by maximum depth, etc

Adaptive Sampling

- Sample only vertices of a space
- Recursive decomposition (cutting) for open spaces, controlled by maximum depth, etc

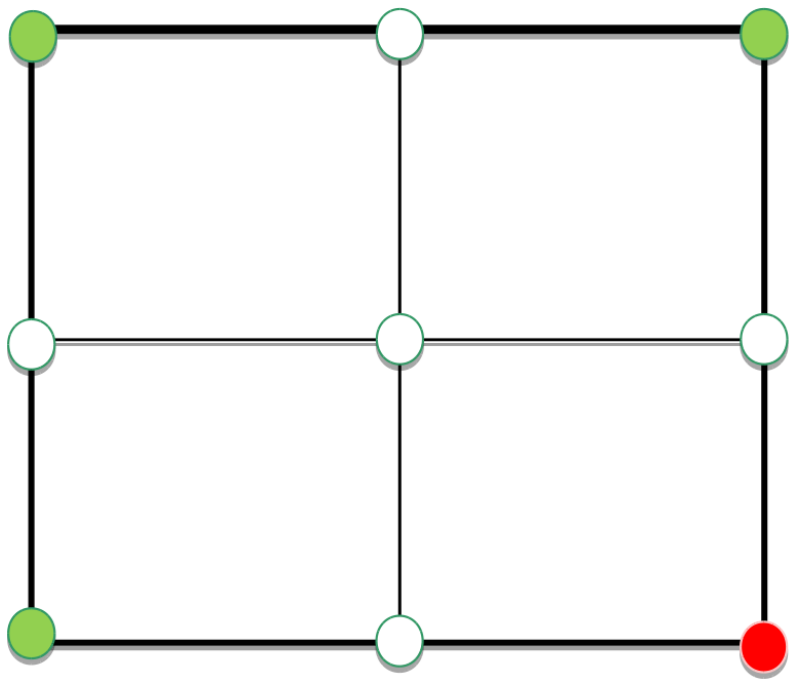
Adaptive Sampling

- Sample only vertices of a space
- Recursive decomposition (cutting) for open spaces, controlled by maximum depth, etc



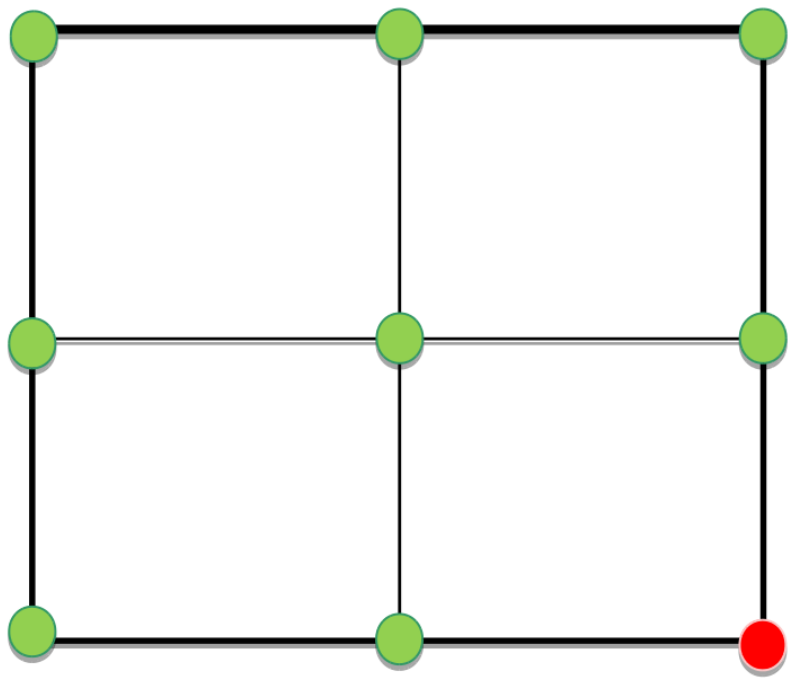
Adaptive Sampling

- Sample only vertices of a space
- Recursive decomposition (cutting) for open spaces, controlled by maximum depth, etc



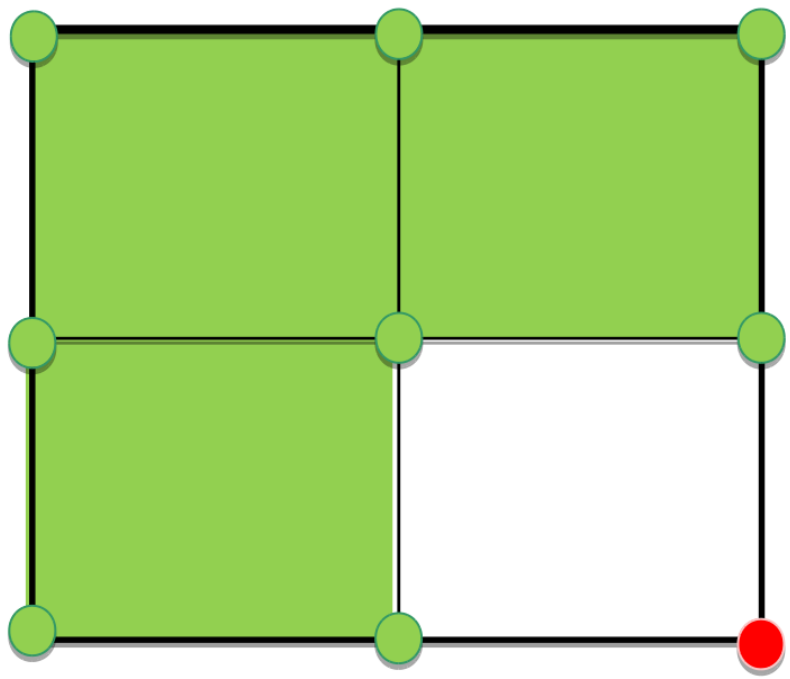
Adaptive Sampling

- Sample only vertices of a space
- Recursive decomposition (cutting) for open spaces, controlled by maximum depth, etc



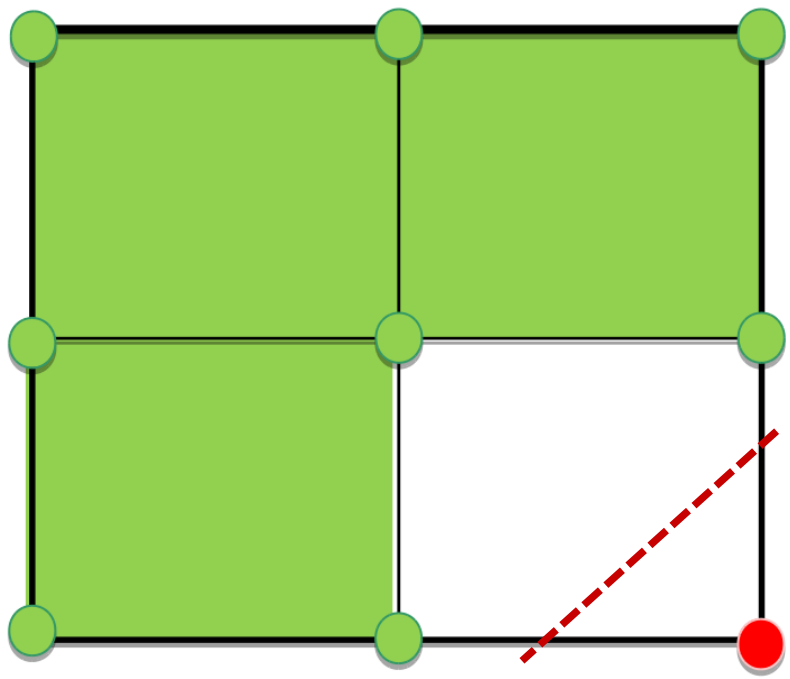
Adaptive Sampling

- Sample only vertices of a space
- Recursive decomposition (cutting) for open spaces, controlled by maximum depth, etc



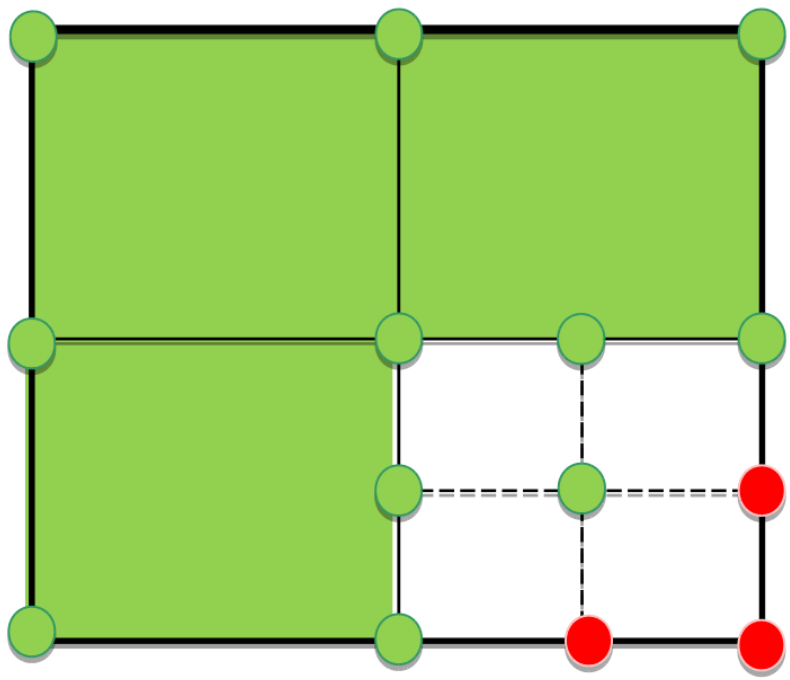
Adaptive Sampling

- Sample only vertices of a space
- Recursive decomposition (cutting) for open spaces, controlled by maximum depth, etc



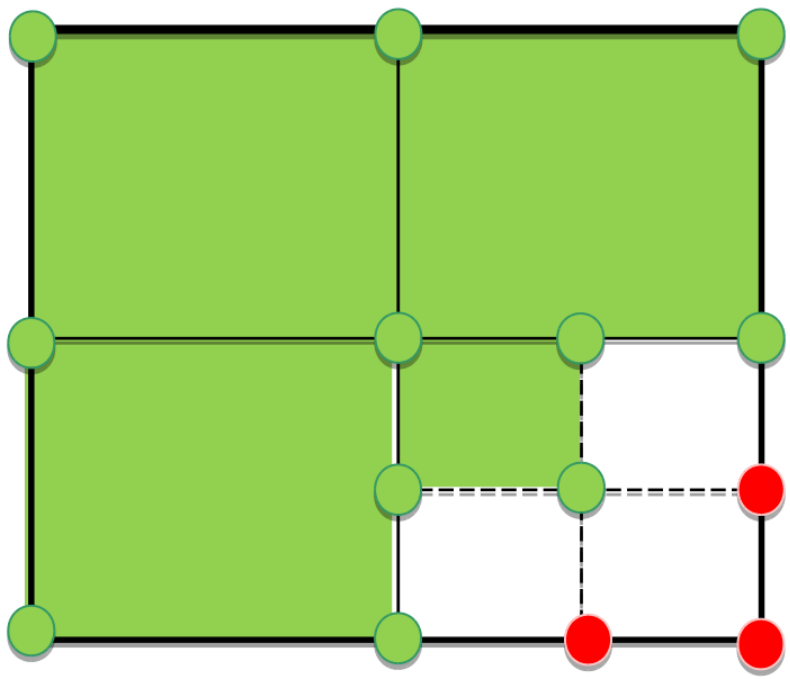
Adaptive Sampling

- Sample only vertices of a space
- Recursive decomposition (cutting) for open spaces, controlled by maximum depth, etc



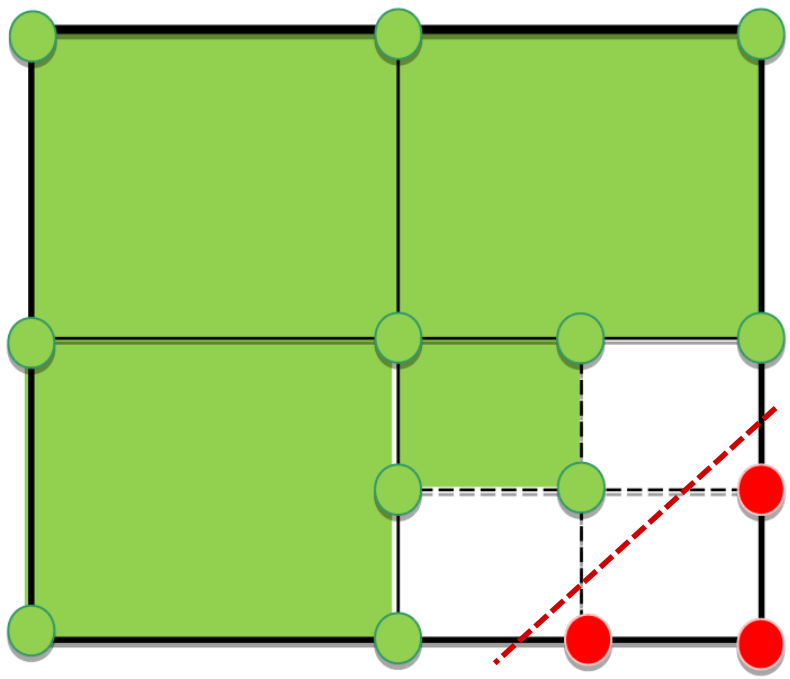
Adaptive Sampling

- Sample only vertices of a space
- Recursive decomposition (cutting) for open spaces, controlled by maximum depth, etc



Adaptive Sampling

- Sample only vertices of a space
- Recursive decomposition (cutting) for open spaces, controlled by maximum depth, etc



Formalization: Runtime context (e.g. Input Size)

- **Context Property Value Space (PVS)** is a n-dimensional finite space
  - A run-time context instance consists of n context property values maps to a point in the PVS
  - PVS has 2^n vertices (corner points)
- Closed and open subspaces
  - Closed subspace: a subspace where a variant wins on all vertices. It heuristically approximate "uninteresting" subspaces
  - Open subspace: otherwise
- Recursive space decomposition

Convexity Assumption from the Observation

- "In real life, the world is smoothly changing, instances close by most of the time have the same labels, and we need not worry about all possible labelings."

  Ethem Alpaydin in "Introduction to Machine Learning, second edition"

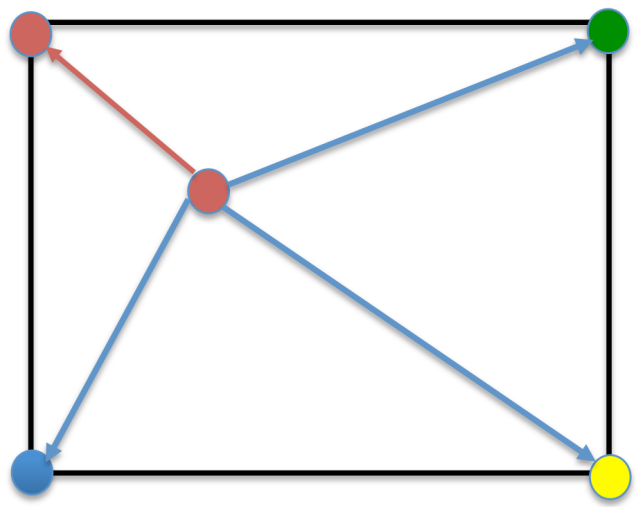
- Our *Convexity Assumption*
  - If all vertices of a subspace share the same winner, then the winner wins in all points of the subspace statistically

Adaptive Sampling In the Big Picture

- Off-line: adaptive sampling, tree construction
- On-line (Run-time)
  - Load the tree
  - Prediction
    - Closed space: look up winner on one vertex
    - Open space: Euclidean-based predictor

Adaptive Sampling In the Big Picture

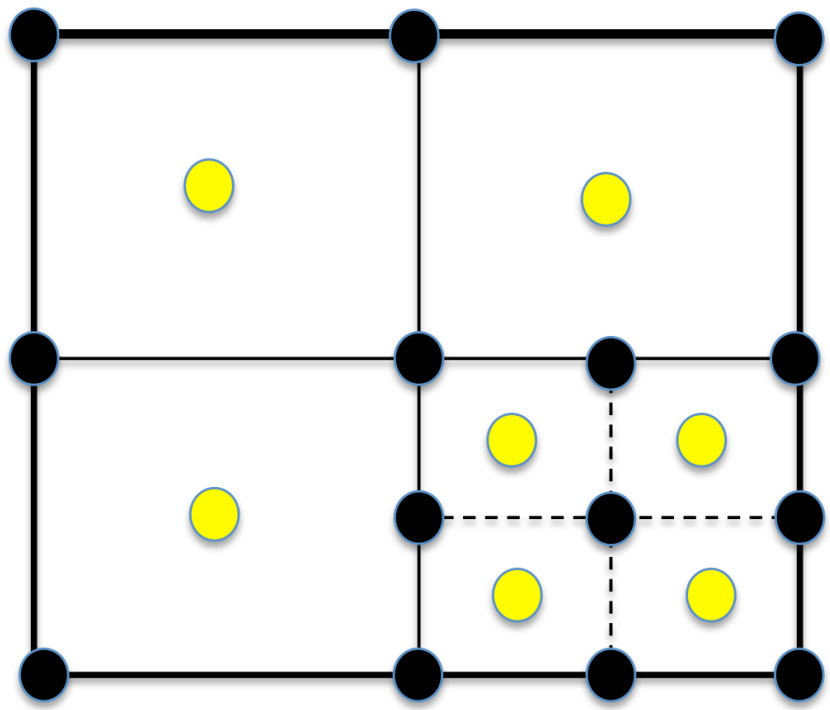
- Off-line: adaptive sampling, tree construction
- On-line (Run-time)
  - Load the tree
  - Prediction
    - Closed space: look up winner on one vertex
    - Open space: Euclidean-based predictor



Techniques for Adaptive Sampling

Light oversampling

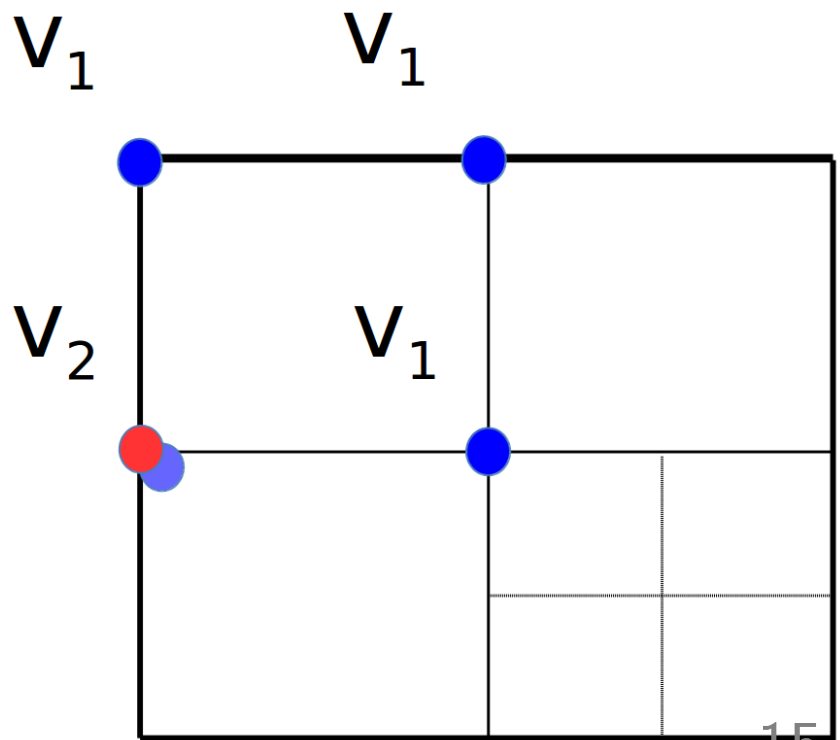
- Sample one extra point in the middle
- Detect holes
- Small increase in overhead
- May increase prediction accuracy



Further Techniques Based on Adaptive Sampling

Thresholding

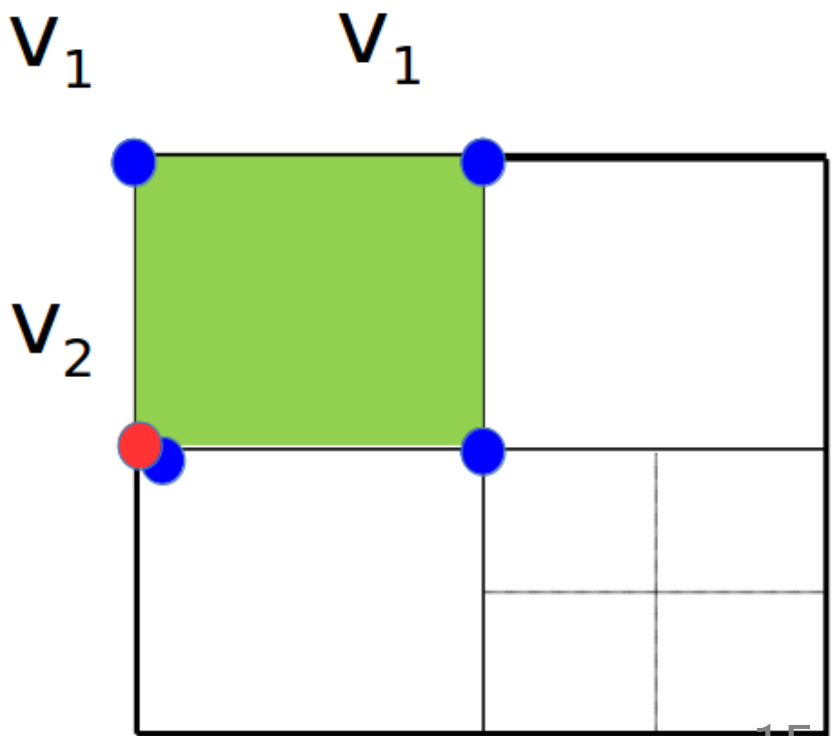
- Relative threshold: $abs(v_i - v_{min})/v_{min} <= \theta$
- Stop splitting early
- Reduce training overhead
- May decrease prediction accuracy



Further Techniques Based on Adaptive Sampling

Thresholding

- Relative threshold: $abs(v_i - v_{min})/v_{min} <= \theta$
- Stop splitting early
- Reduce training overhead
- May decrease prediction accuracy



Further Techniques Based on Adaptive Sampling

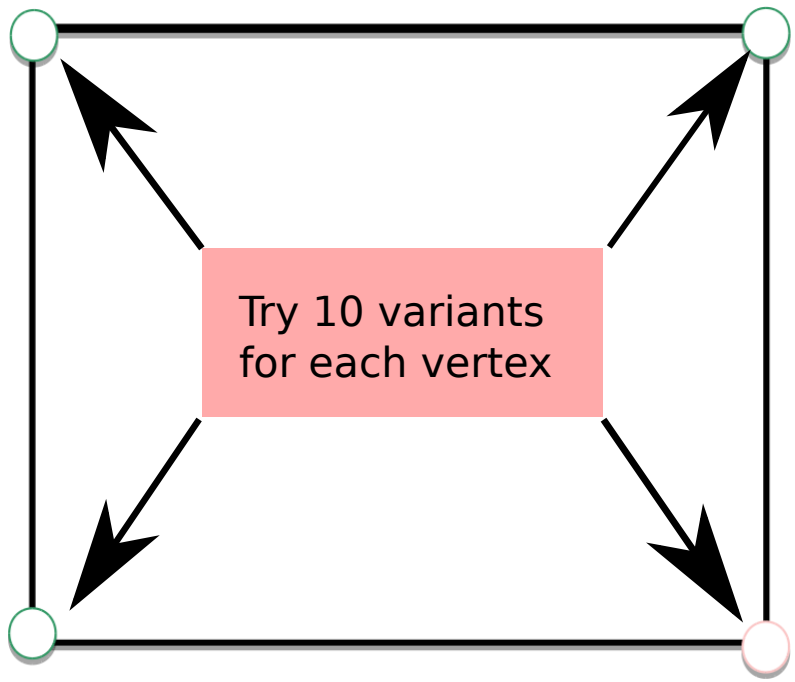
Implementation Pruning

- Only winners of the vertices will involve in the future sampling of the subspace
- Reduce overhead remarkably if many implementation variants are available for an interface
- May lead to loss of optimization potential

Further Techniques Based on Adaptive Sampling

Implementation Pruning

- Only winners of the vertices will involve in the future sampling of the subspace
- Reduce overhead remarkably if many implementation variants are available for an interface
- May lead to loss of optimization potential



Further Techniques Based on Adaptive Sampling

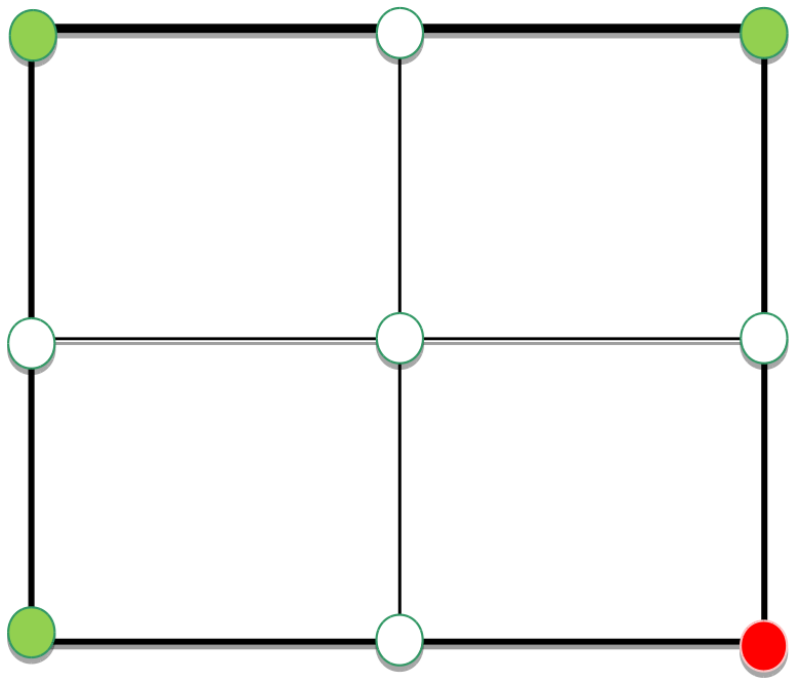
Implementation Pruning

- Only winners of the vertices will involve in the future sampling of the subspace
- Reduce overhead remarkably if many implementation variants are available for an interface
- May lead to loss of optimization potential

Further Techniques Based on Adaptive Sampling

Implementation Pruning

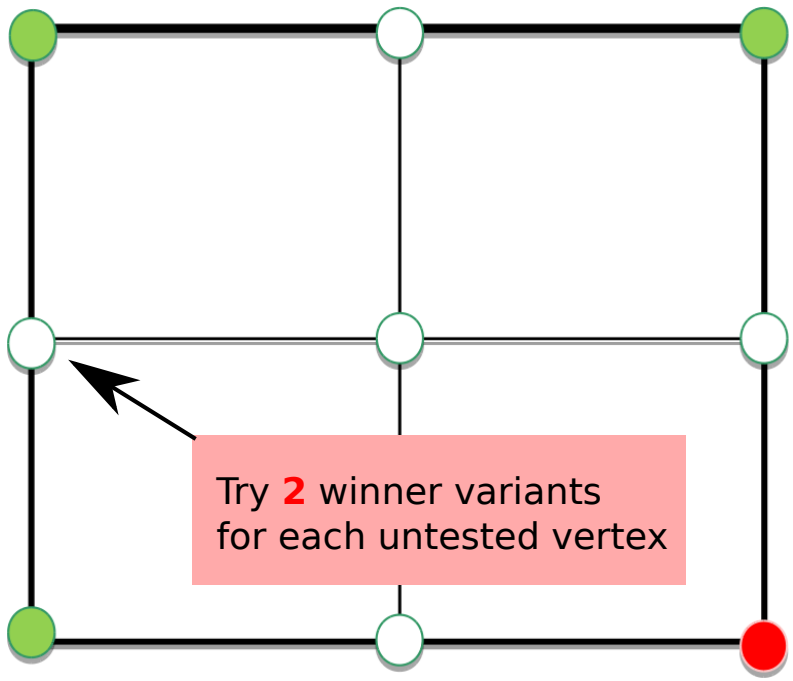
- Only winners of the vertices will involve in the future sampling of the subspace
- Reduce overhead remarkably if many implementation variants are available for an interface
- May lead to loss of optimization potential



Further Techniques Based on Adaptive Sampling

Implementation Pruning

- Only winners of the vertices will involve in the future sampling of the subspace
- Reduce overhead remarkably if many implementation variants are available for an interface
- May lead to loss of optimization potential



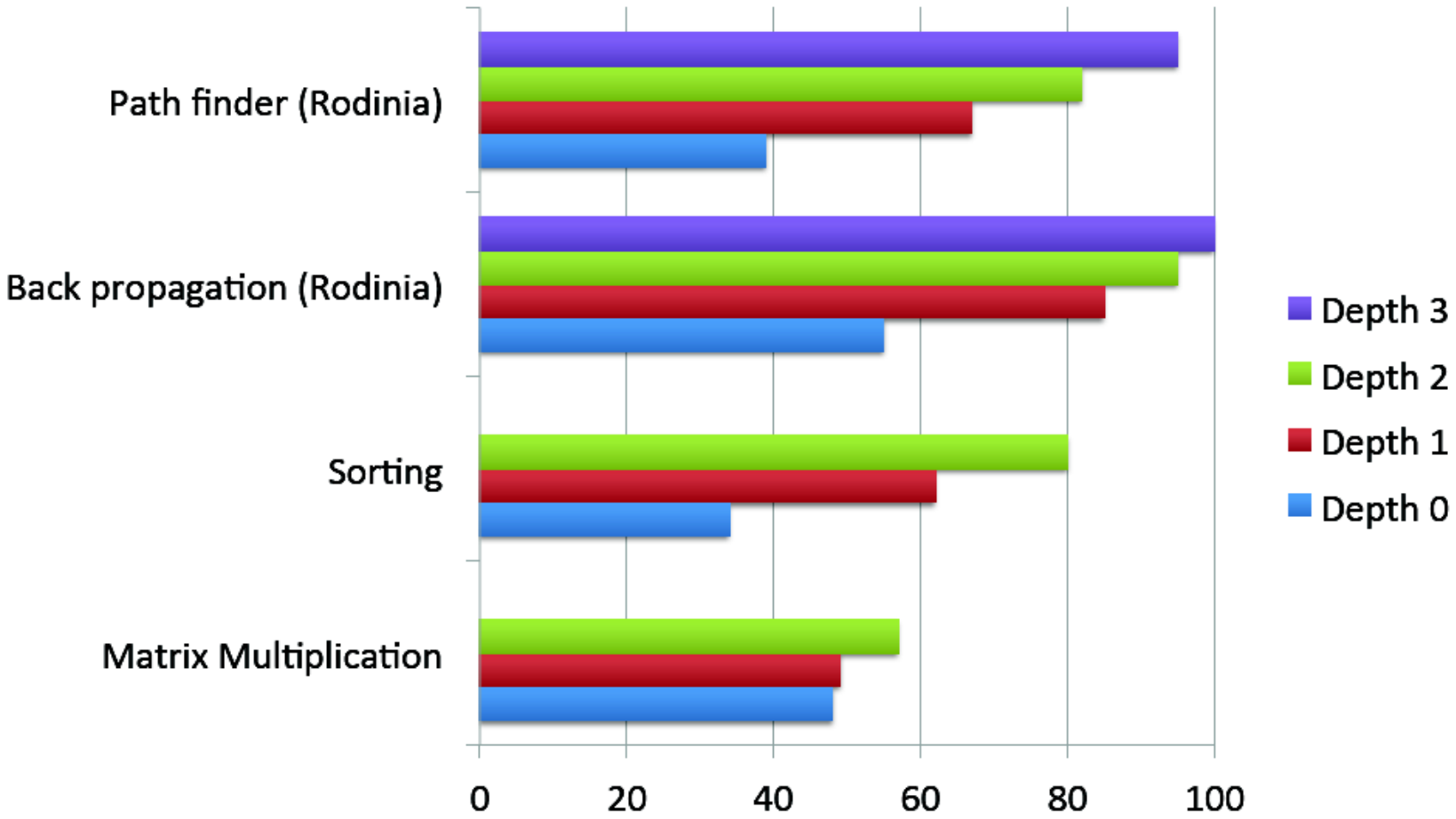
Platform

Name	CPU	GPU	OS	Compiler
Cora	16 Intel(R) Xeon(R) CPU X5550 @ 2.67GHz	3 GPUs: two nVidia Tesla C2050 and one Tesla C1060	RHEL 5.6	gcc 4.1.2 and nvcc V0.2.1221
Fermi	8 Intel(R) Xeon(R) CPU E5520 @ 2.27GHz	two Tesla M2050 GPUs	3.2.1-2-ARCH	gcc 4.6.2 and nvcc V0.2.1221

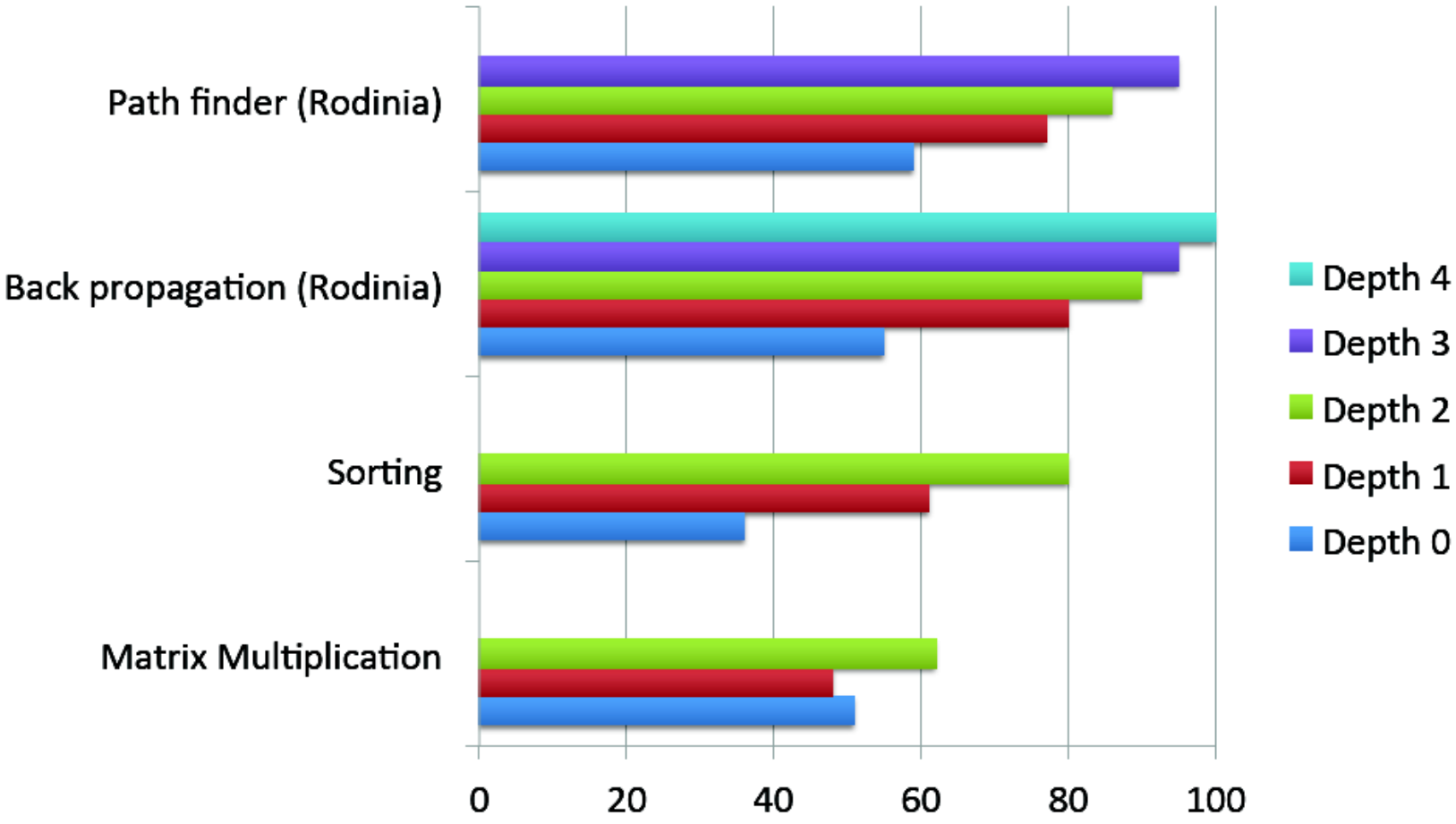
Benchmarks

Benchmark	Feature modeling	Range	Implementation variants
Matrix-matrix multiplication (MM)	row size, column size of first matrix, column size of second matrix	(30, 30, 30) to (300, 300, 300)	Sequential implementation and a variant by loop rearrangement, CUDA impl., BLAS impl., Pthread impl. and five of its variants from loop rearrangement
Sorting (ST)	array size; discretization of array values distribution (sampled number of inversions)	(1,0) to (10k,10)	bubble sort, insertion sort, merge sort, quick sort
Path finder (PF)	row; column	(1,1) to (10k,20k)	OpenMP implementation, CUDA implementation
Backpropagation (BP)	array size	(1k) to (100k)	OpenMP implementation, CUDA implementation

Prediction Accuracy (%) on Cora: Base-line adaptive off-line sampling



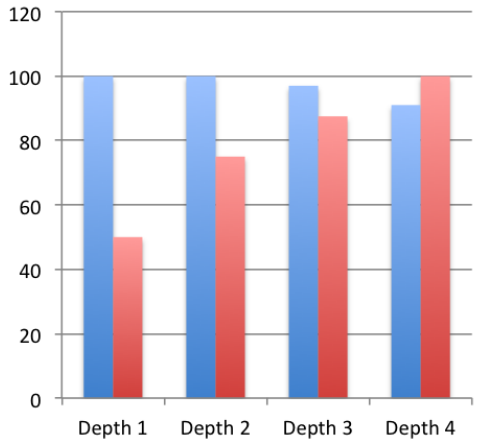
Prediction Accuracy (%) on Fermi: Base-line adaptive off-line sampling



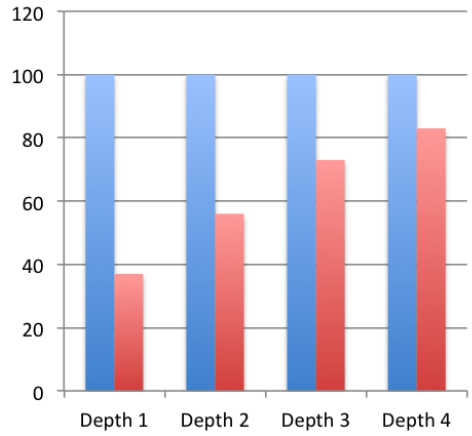
Other Metrics

- Absolute runtime overhead: 4 - 23 μs
- Relative runtime overhead: 0.2%
- Average sampling rate: 0.053%

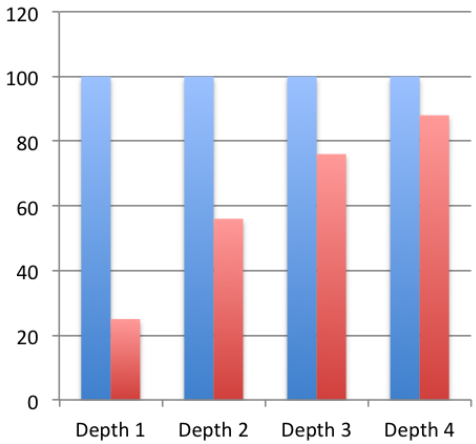
Test against Convexity Assumption



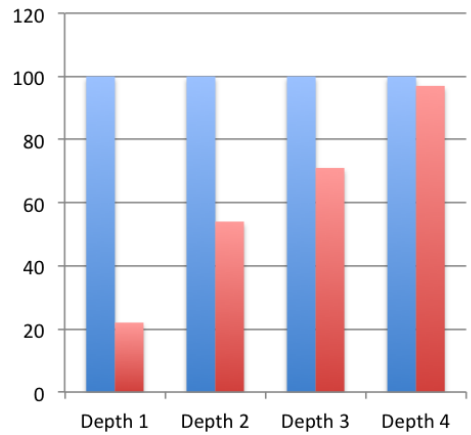
BP



MM



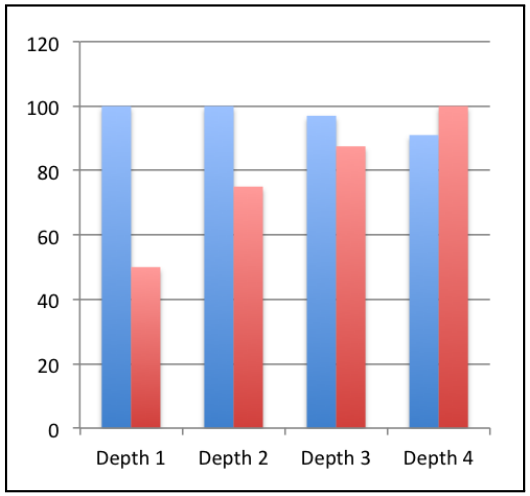
PF



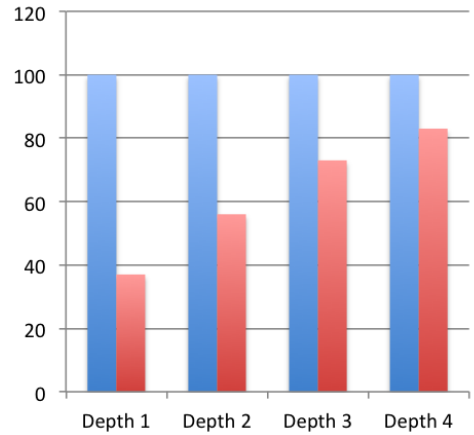
ST

Accuracy(%) on closed space. Pecentage(%) on closed space.

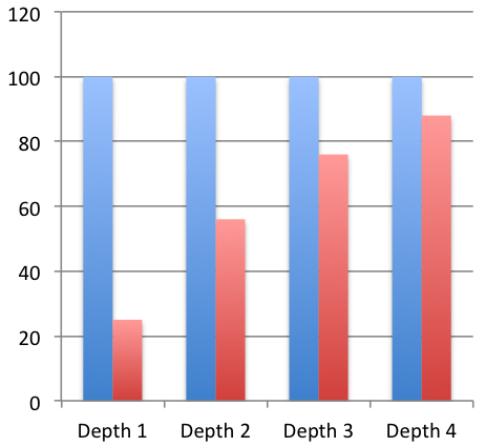
Test against Convexity Assumption



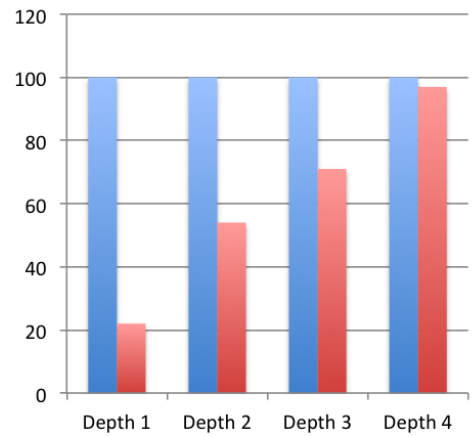
BP



MM



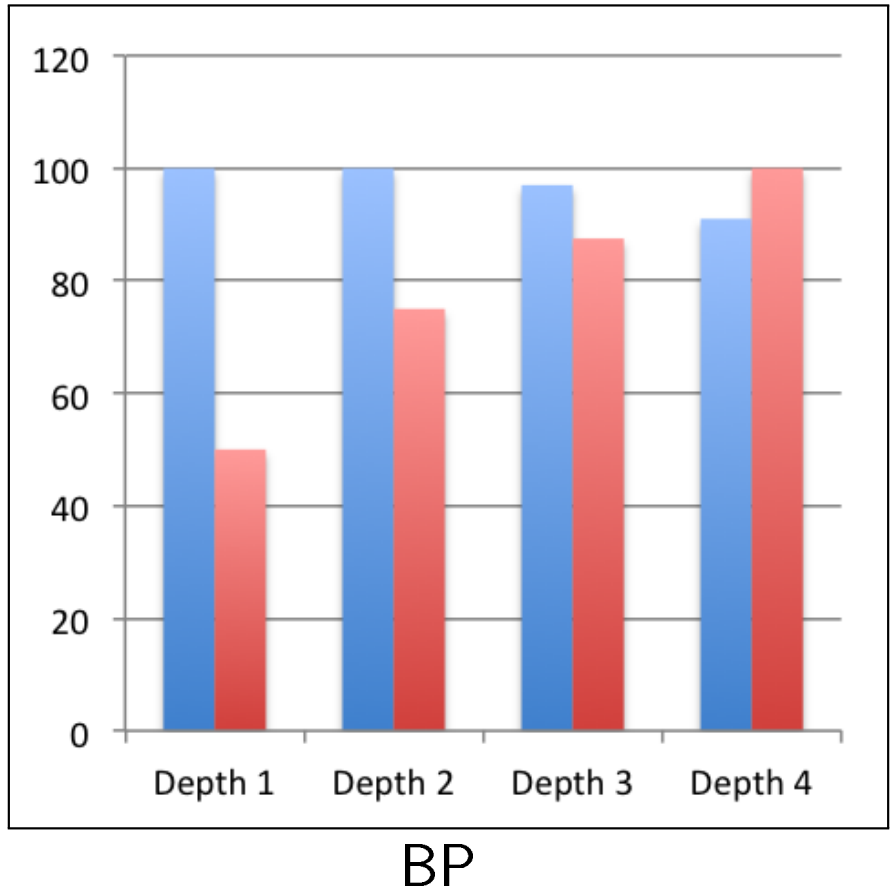
PF



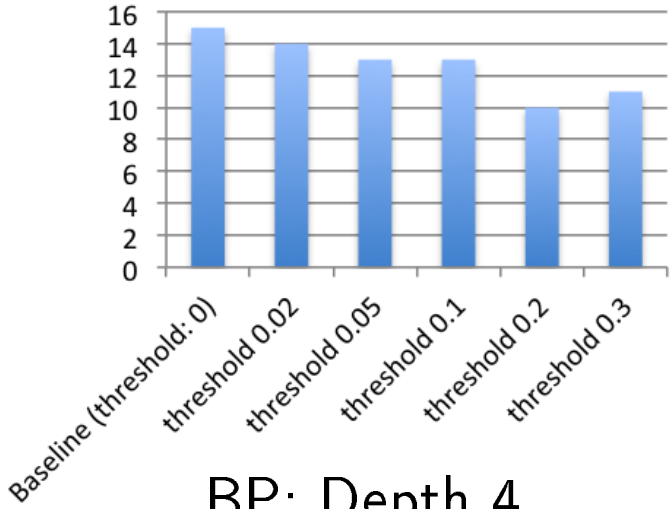
ST

Accuracy(%) on closed space. Pecentage(%) on closed space.

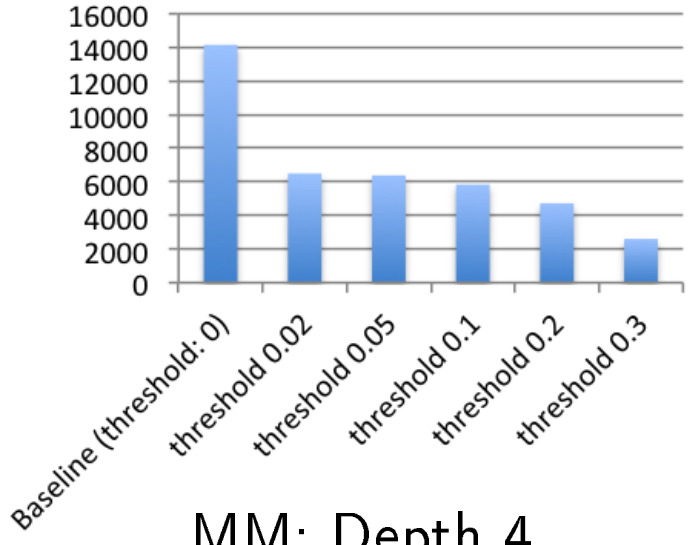
Test against Convexity Assumption



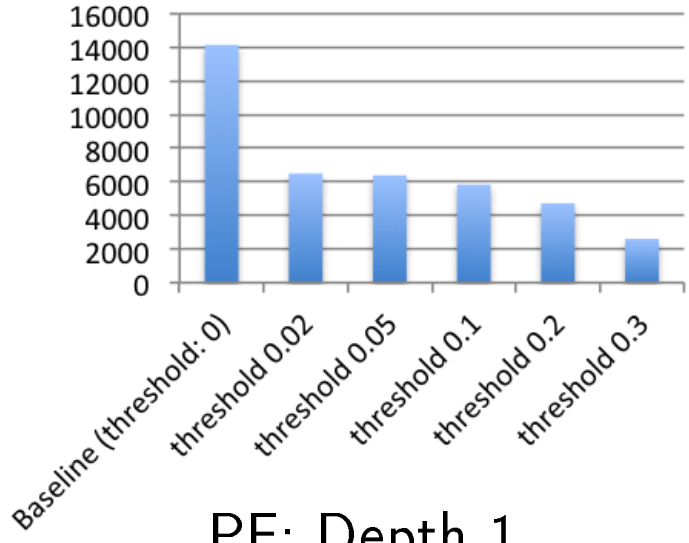
Thresholding Effect: Training Time



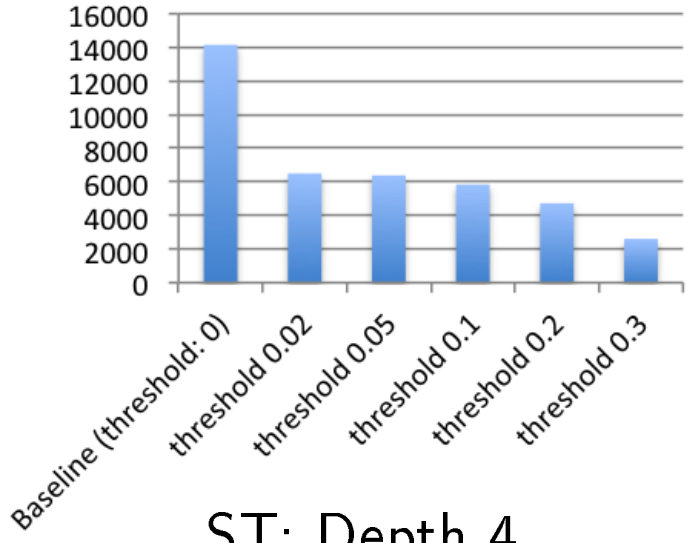
BP: Depth 4



MM: Depth 4

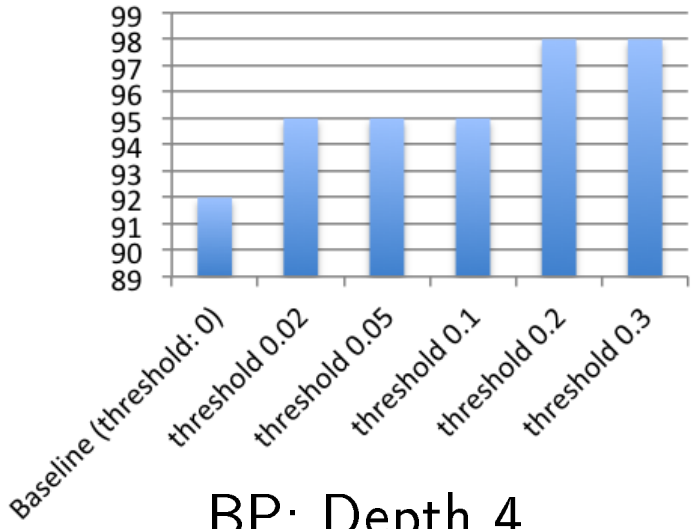


PF: Depth 1

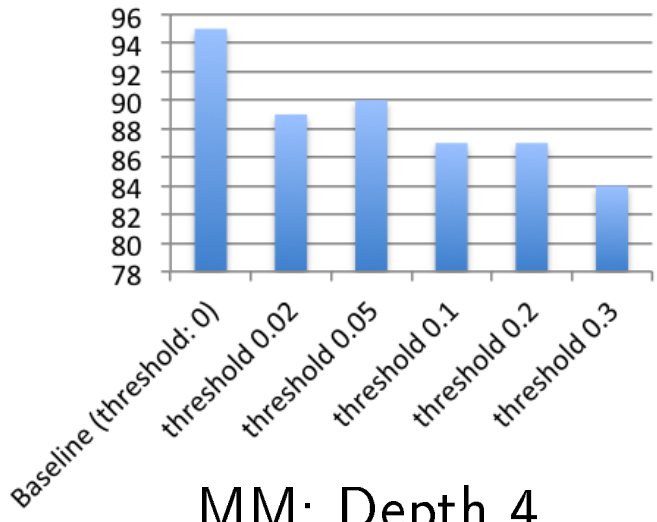


ST: Depth 4

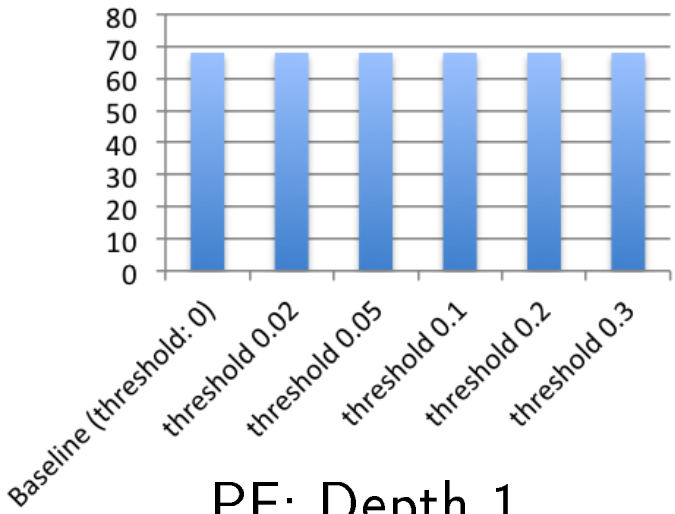
Thresholding Effect: Prediction Accuracy



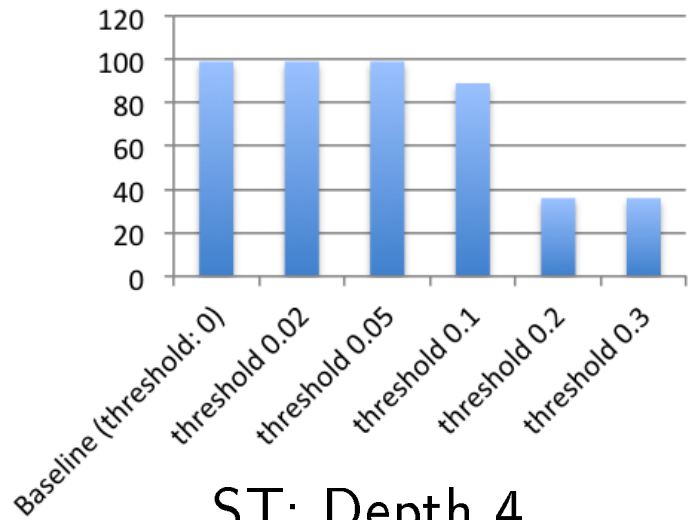
BP: Depth 4



MM: Depth 4

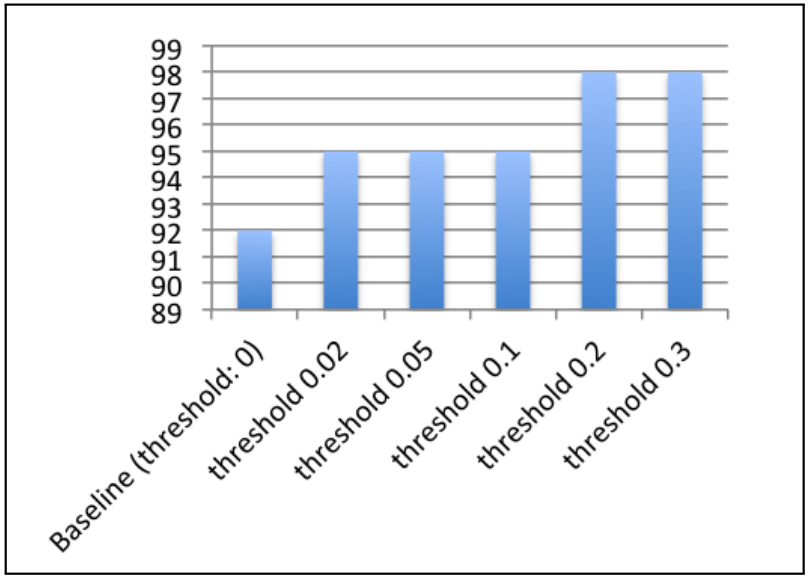


PF: Depth 1

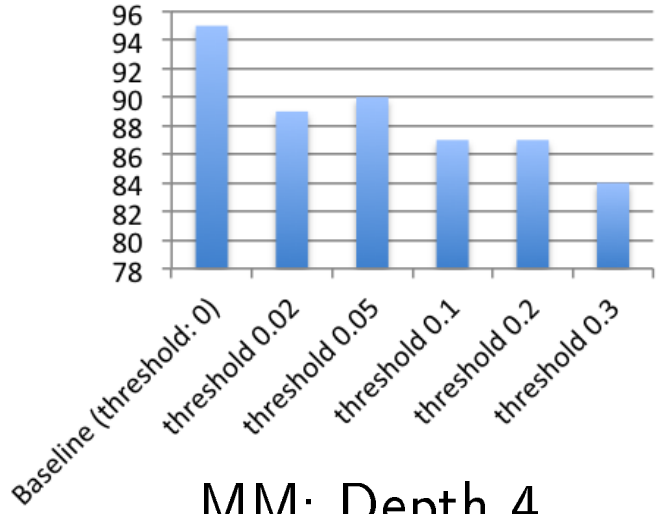


ST: Depth 4

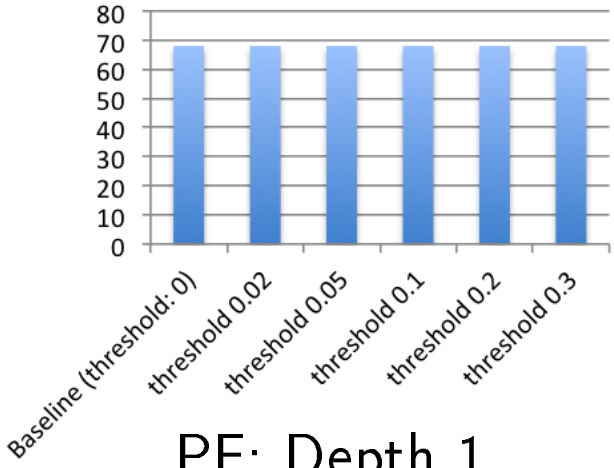
Thresholding Effect: Prediction Accuracy



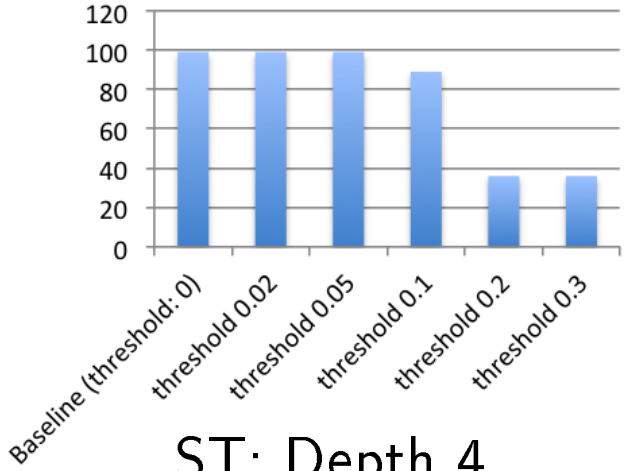
BP: Depth 4



MM: Depth 4

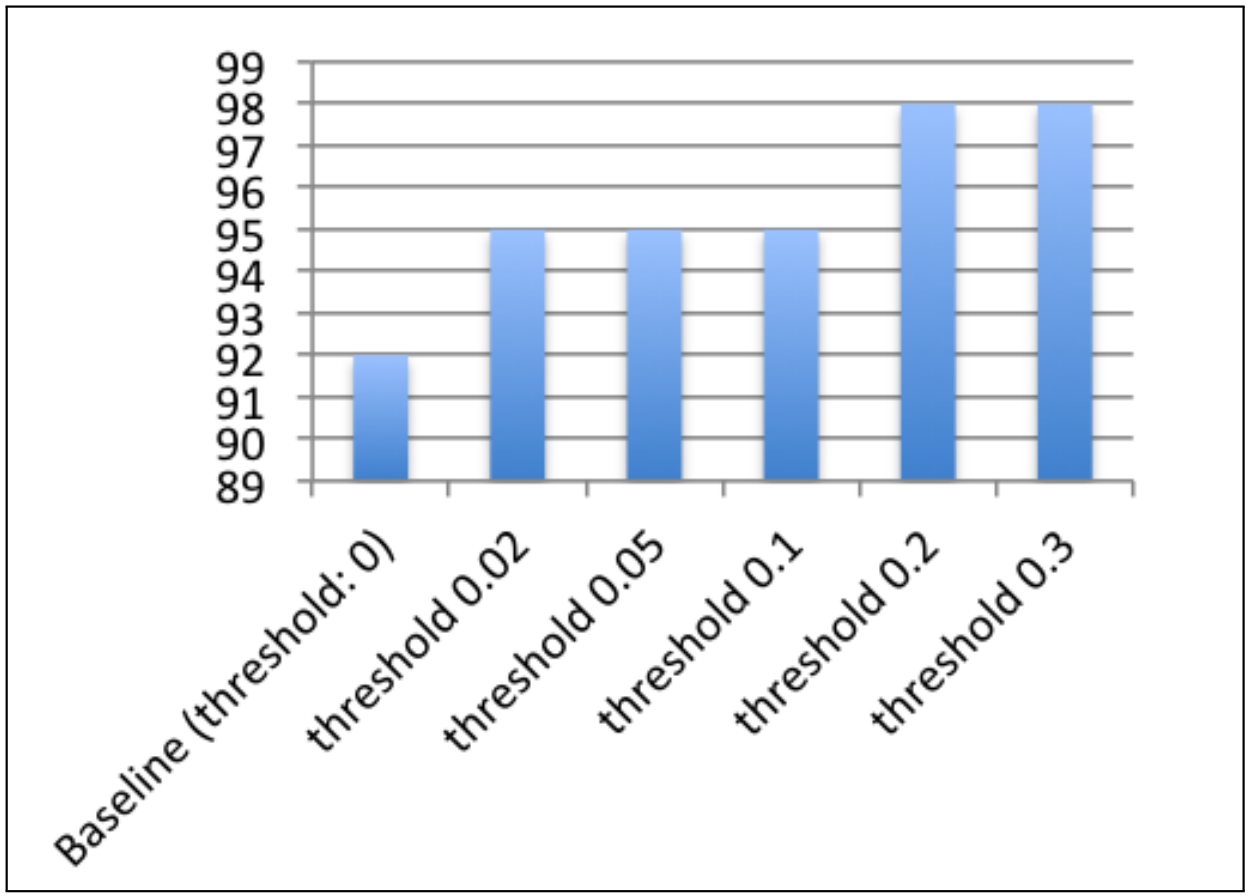


PF: Depth 1



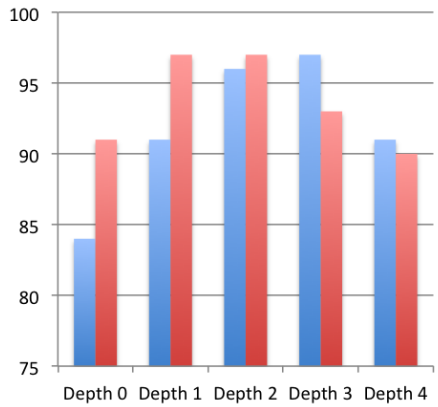
ST: Depth 4

Thresholding Effect: Prediction Accuracy

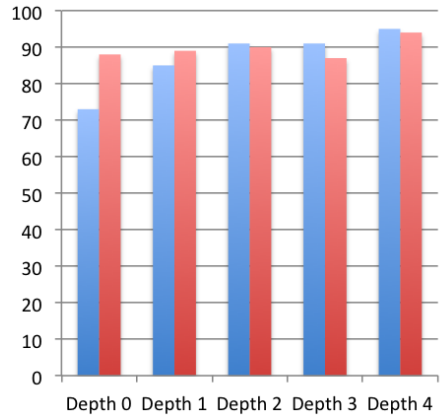


BP: Depth 4

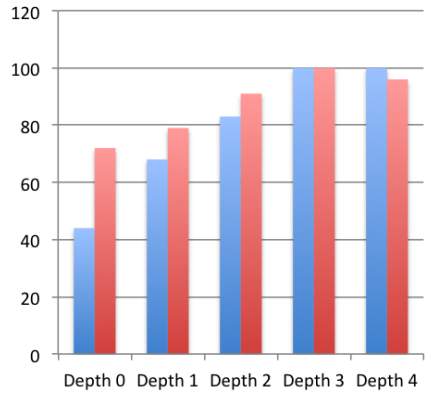
Oversampling Effect: Prediction Accuracy



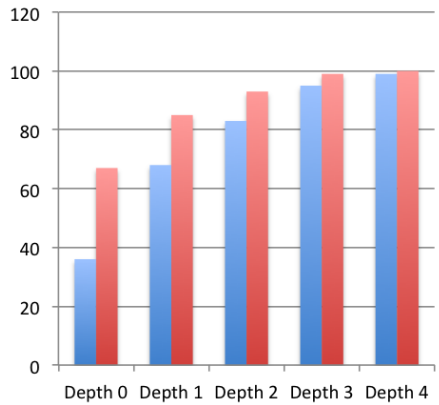
BP



MM



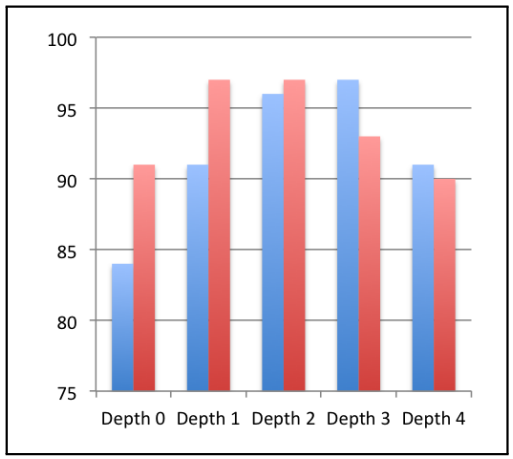
PF



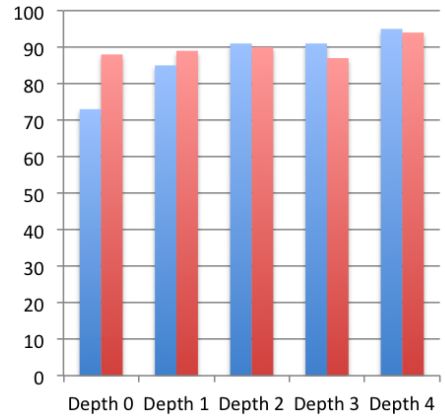
ST

Adaptive Sampling. Adaptive Sampling with light oversampling.

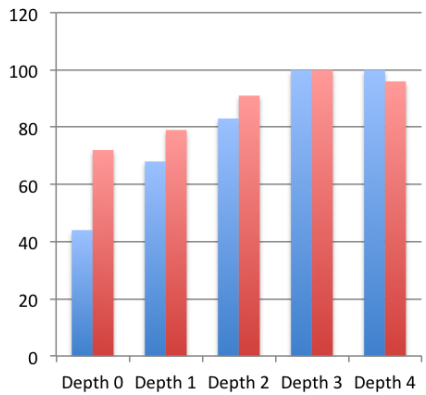
Oversampling Effect: Prediction Accuracy



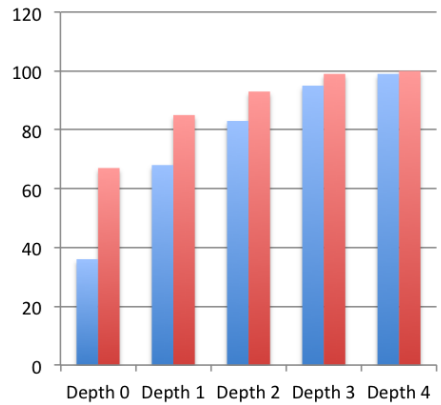
BP



MM



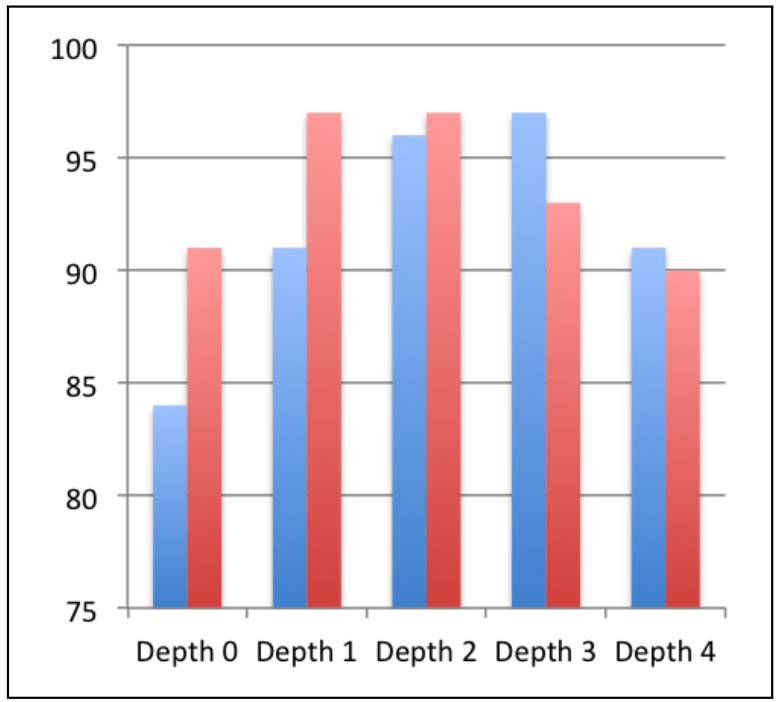
PF



ST

Adaptive Sampling. Adaptive Sampling with light oversampling.

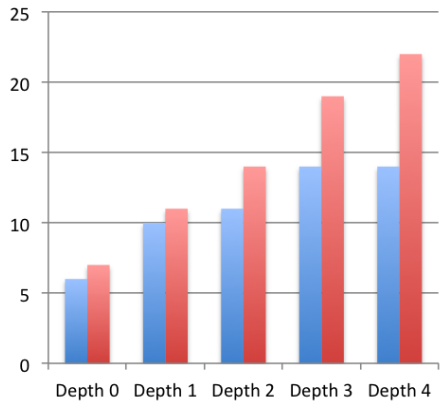
Oversampling Effect: Prediction Accuracy



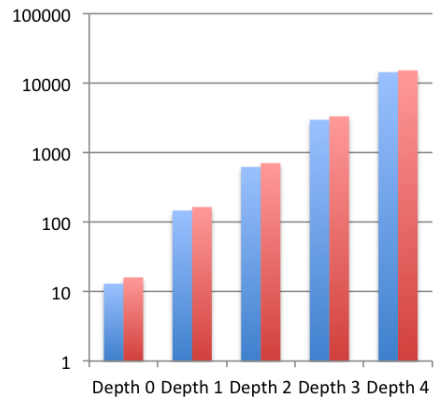
BP

Adaptive Sampling. Adaptive Sampling with light oversampling.

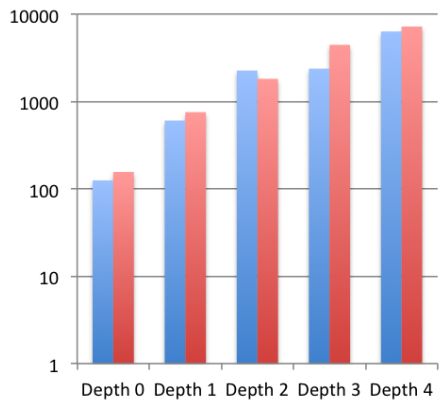
Oversampling Effect: Training Time



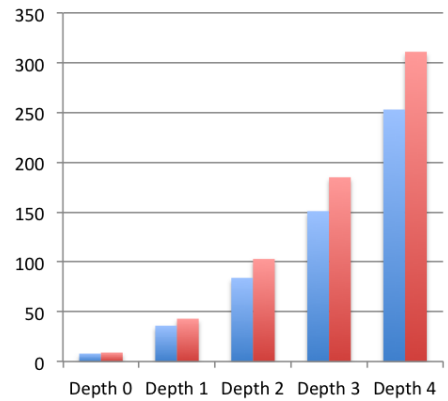
BP



MM



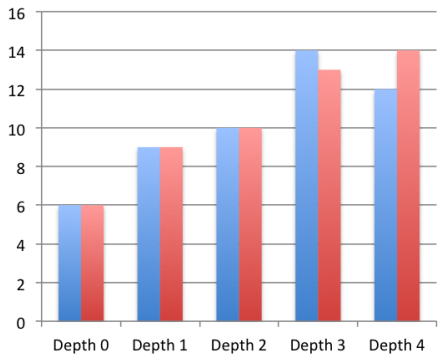
PF



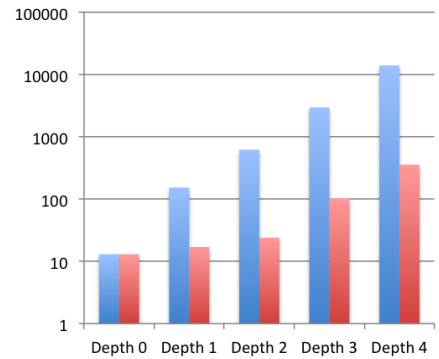
ST

Adaptive Sampling. Adaptive Sampling with light oversampling.

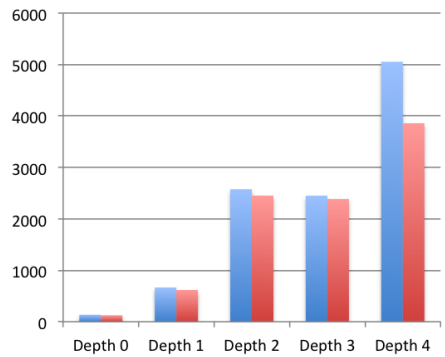
Implementation Pruning Effect: Training Time



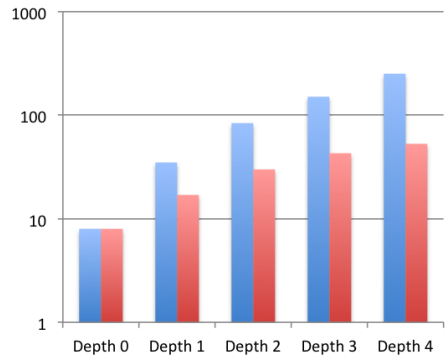
BP



MM



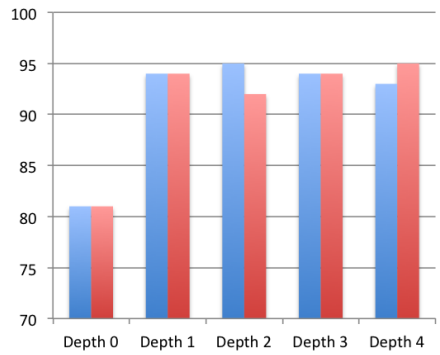
PF



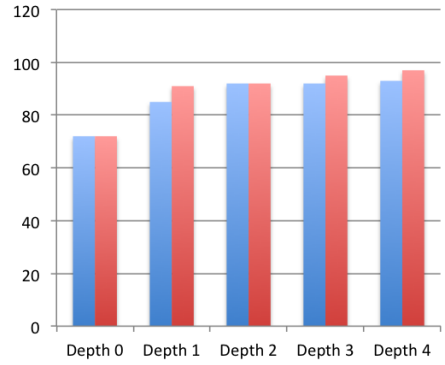
ST

Adaptive Sampling. Adaptive Sampling with impl pruning.

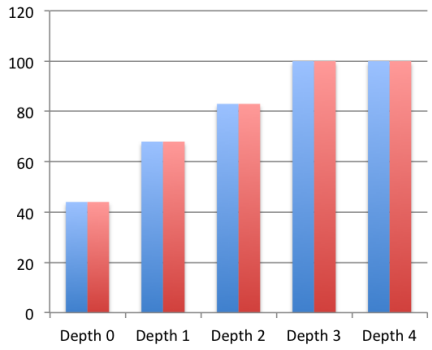
Implementation Pruning Effect: Prediction Accuracy



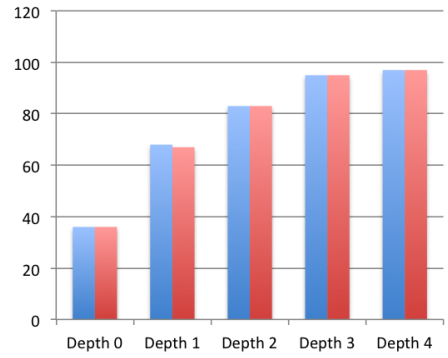
BP



MM



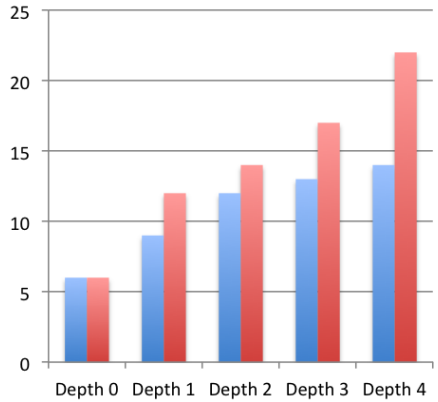
PF



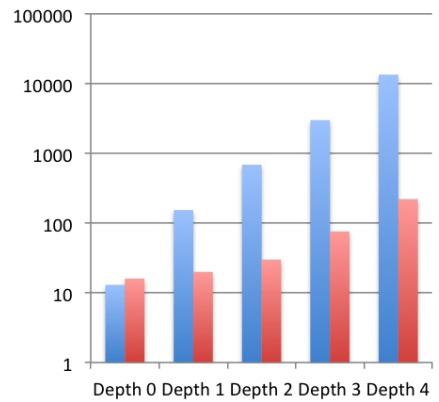
ST

Adaptive Sampling. Adaptive Sampling with impl pruning.

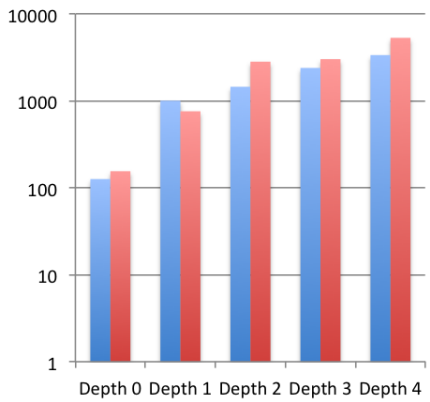
Combo Effect: Training Time



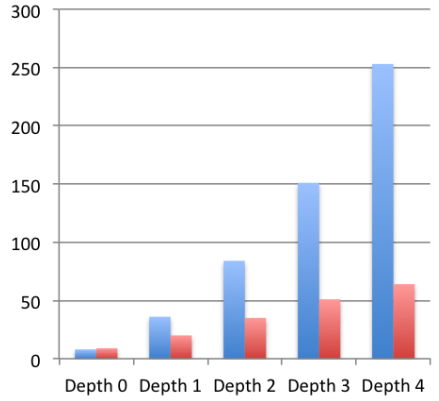
BP



MM



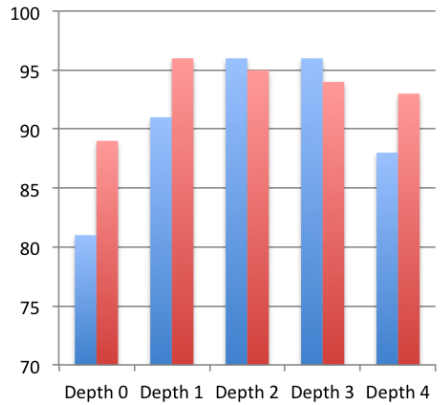
PF



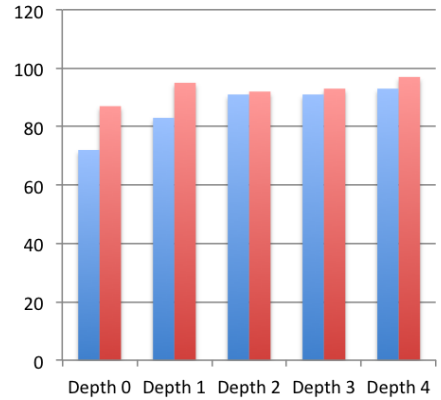
ST

Adaptive Sampling. Adaptive Sampling with the combo of oversampling and impl pruning.

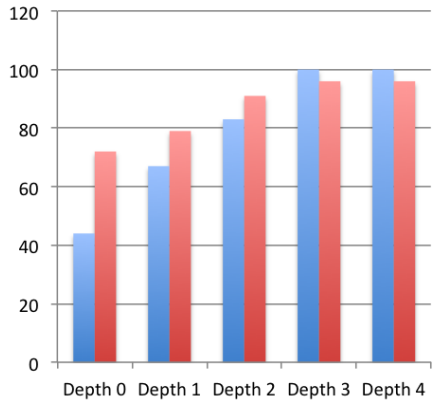
Combo Effect: Prediction Accuracy



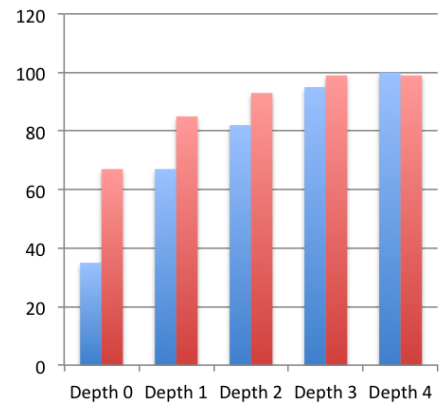
BP



MM



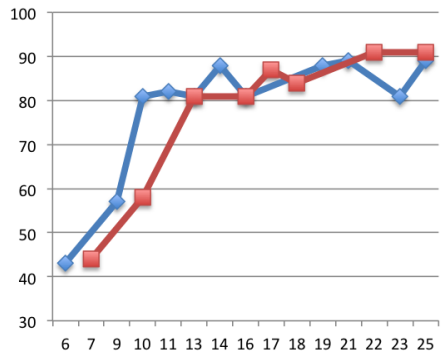
PF



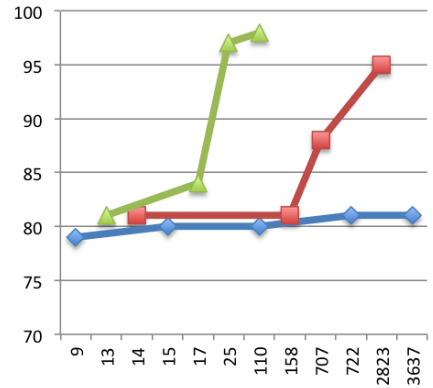
ST

Adaptive Sampling. Adaptive Sampling with combo of oversampling and impl pruning.

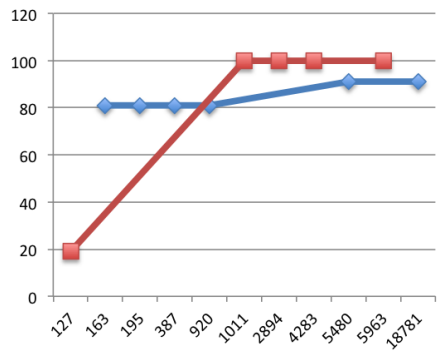
Adaptive Sampling vs Random Sampling



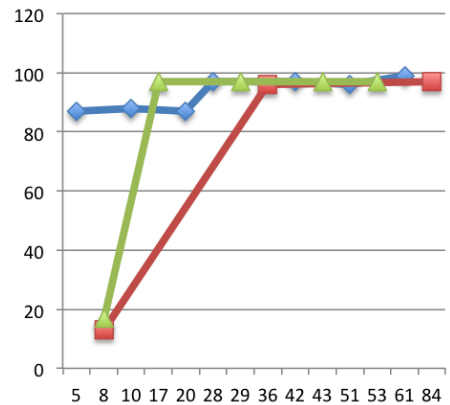
BP



MM



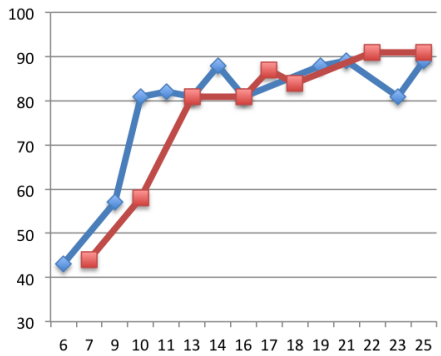
PF



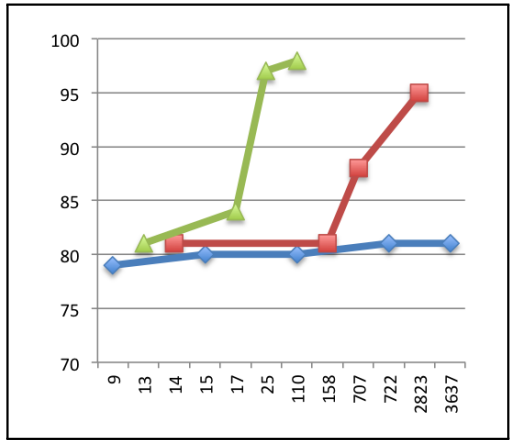
ST

Random Sampling Adaptive Sampling Adaptive Sampling with the combo of oversampling and impl pruning.

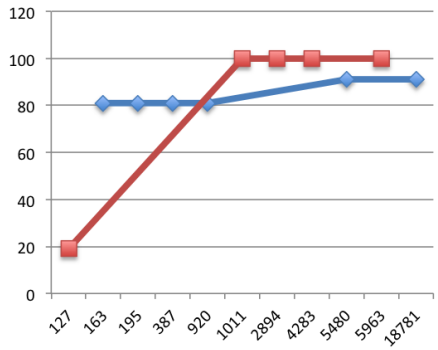
Adaptive Sampling vs Random Sampling



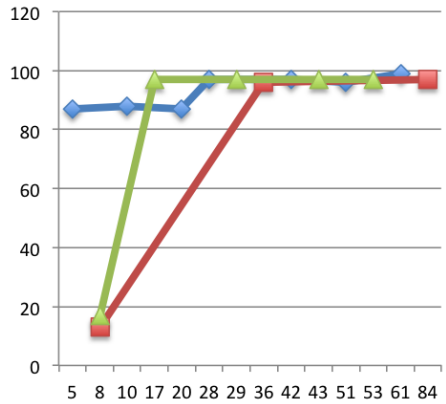
BP



MM



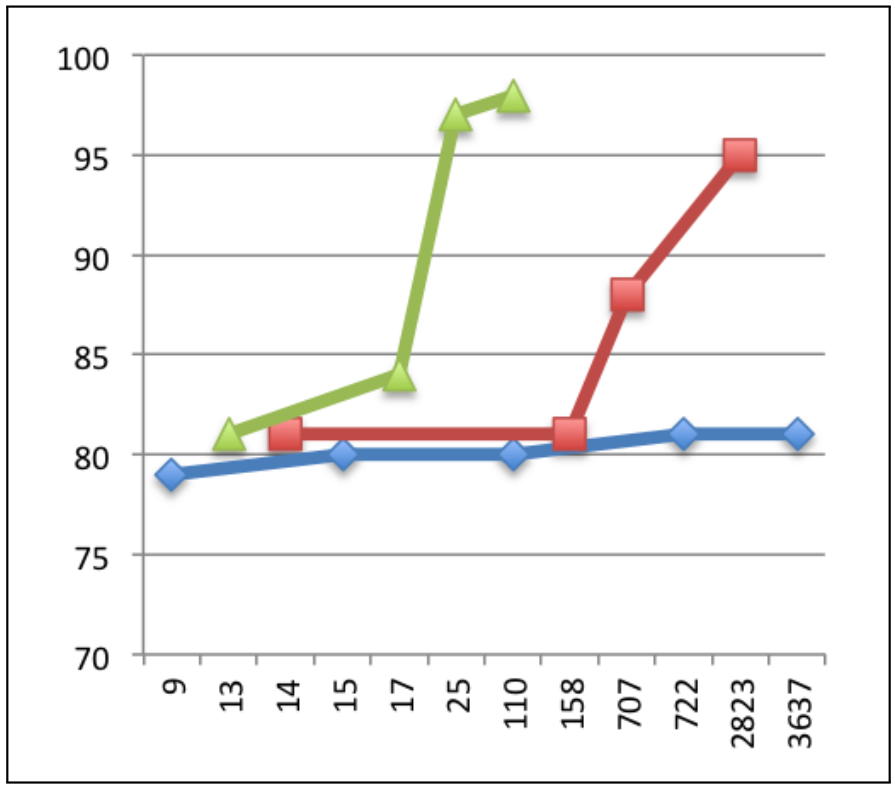
PF



ST

Random Sampling Adaptive Sampling Adaptive Sampling with the combo of oversampling and impl pruning.

Adaptive Sampling vs Random Sampling



Random Sampling Adaptive Sampling Adaptive Sampling with the combo of oversampling and impl pruning.

Conclusion

- Convexity Assumption holds for all benchmarks used, more to test
- Three techniques help to decrease training time or increase prediction accuracy
  - Thresholding
  - Light Oversampling
  - Implementation pruning
- The right combination can combine the advantages
- Adaptive sampling shows benefits against random sampling

More questions please contact me: lu.li@liu.se, +46704759692